Shivaansh Kapoor
Email: shivaansh@ucsb.edu
Perm: 8542821

Architecture:

The architecture of my program consists of a few key components. Instead of using a class, I used functions in order to implement the Naive Bayes Classifier. I have four main functions that make up my program: *makeDatasets, calculateProbability, classifier, and train_and_test.* The *makeDatasets* function is used at the beginning of the program to parse the text files and return pandas datasets. The *calculateProbability* function is used to calculate the probability of a specific value in a Gaussian distribution given the mean and standard deviation. The *classifier* function is what is actually called in order to classify a row in the testing file. It returns either 0 or 1 based on its prediction of if it will rain tomorrow or not. Lastly, the *train_and_test* function acts as the main function for the program, as it organizes the datasets and calls the *classifier* function for each row in the testing file.

Preprocessing:

The preprocessing part of my program is taken care of using the *makeDatasets* and the *train_and_test* functions. Initially, the *train_and_test* function calls the *makeDatasets* function. The *makeDatasets* function takes in two parameters (the file path for the training data as parameter 1 and the file path of the testing data as parameter 2). It then reads the data from each of the files into a pandas dataframe using *pandas.read_csv.* Once we have the training and testing dataframes, we split the training dataframe into two (1 for which RainTomorrow is true and 1 for which RainTomorrow is false). Then it returns the three dataframes (train_dataset_yes, train_dataset_no, test_dataset). Then, the *train_and_test* function finds the means and standard deviations of the numerical columns of train_dataset_yes and train_dataset_no. Now, the program is ready for classification.

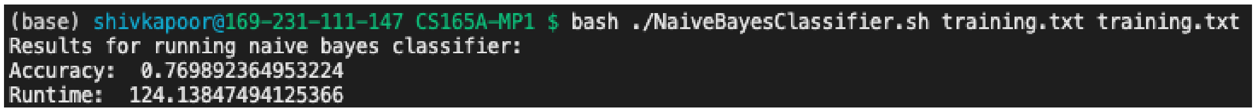
Model Building:

Once the preprocessing is done, the *train_and_test* function calls the *classifier* function for each row of the testing dataset. The *classifier* function takes 7 arguments (train_dataset_yes, train_dataset_no, test_row, means_yes, stds_yes, means_no, stds_no). It then initializes 2 variables (c_yes and c_no) to $log_{10}(1/2)$ (since half the rows of the training data have RainTomorrow as yes and the other half are no). These two variables will keep track of the probabilities for the rest of the function. Then, we

analyze each of the categorical columns of the test row. We do this by finding the counts of the values in the categorical columns of the test row for train_dataset_yes and train_dataset_no and add $log_{10}(countYes/totalYes)$ to c_yes and $log_{10}(countNo/totalNo)$ to c_no. Now that the categorical values have been taken care of, we move on to the numerical columns. For each numerical column, we find the probability of the value based on the gaussian distribution for that column for the train_dataset_yes and train_dataset_no datasets. Then we add the log of the respective probabilities to c_yes and c_no. Now that we have gone through all the columns, we can classify the row as if it's going to rain tomorrow or not. We do this by comparing c_yes and c_no. If c_yes is
greater than c_no, we return 1, else we return 0. Then, the *train_and_test* function prints the value returned from the *classifier* function.

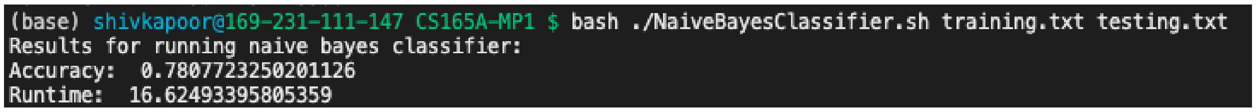
Results:

Results for training naive bayes classifier with training.txt and testing it on training.txt:



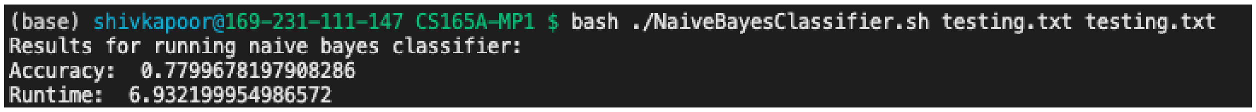
```
(base) shivkapoor@169-231-111-147 CS165A-MP1 $ bash ./NaiveBayesClassifier.sh training.txt training.txt
Results for running naive bayes classifier:
Accuracy:  0.769892364953224
Runtime:  124.13847494125366
```

Results for training naive bayes classifier with training.txt and testing it on testing.txt:



```
(base) shivkapoor@169-231-111-147 CS165A-MP1 $ bash ./NaiveBayesClassifier.sh training.txt testing.txt
Results for running naive bayes classifier:
Accuracy:  0.7807723250201126
Runtime:  16.62493395805359
```

Results for training naive bayes classifier with testing.txt and testing it on testing.txt:



```
(base) shivkapoor@169-231-111-147 CS165A-MP1 $ bash ./NaiveBayesClassifier.sh testing.txt testing.txt
Results for running naive bayes classifier:
Accuracy:  0.7799678197908286
Runtime:  6.932199954986572
```

Challenges:

I ran into one main challenge when writing the naive bayes classifier. I had never done naive bayes classification with numerical data and categorical data, so using both of them together was a challenge. My solution was to split the categorical and numerical columns and calculate their probabilities individually. I used the count method taught in class for categorical features and assumed a Gaussian distribution for the numerical features. I first calculated the probabilities for categorical features and then moved on to numerical features. This worked well and I got a fairly high accuracy for all my tests.

Weaknesses:

Even though my classifier works well, it can definitely be improved. The main weakness in my program is that I use all the features given to calculate the final probabilities. With some trial and error, I can figure out which features do not really help with classification and remove them to improve the overall accuracy of the classifier.